INFO 6250 Final Project Report

Title: KSearch – Local search service website

Author: Jixiao Yang

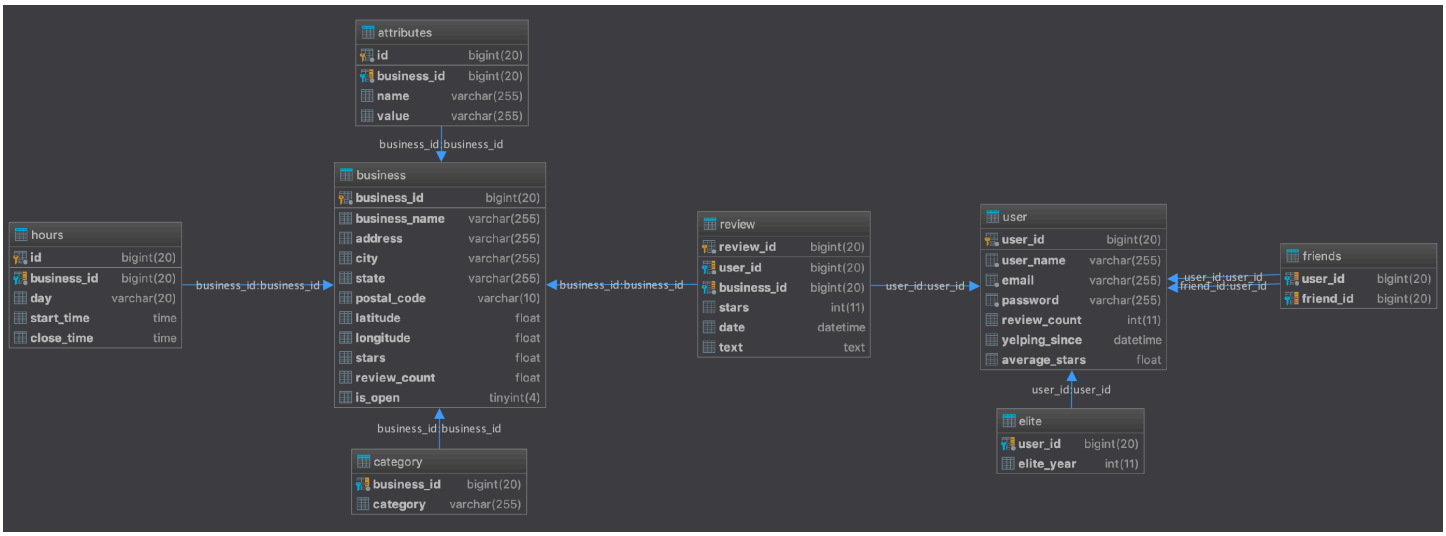
1. Project Summary

Design a website that can let people search for local services. People can search their preferred services in this system and check their rating score and text comments. According to different services, people can check their basic information and rate score and writing text comments to them. What's more, people can check their account basic information and build their friend network.

2. Technologies

Database: MySQL
Design Pattern: MVC
Backend Frameworks: Spring 4.3.1. Release, Hibernate 5.4.1. Final
Frontend Frameworks: Bootstrap 4

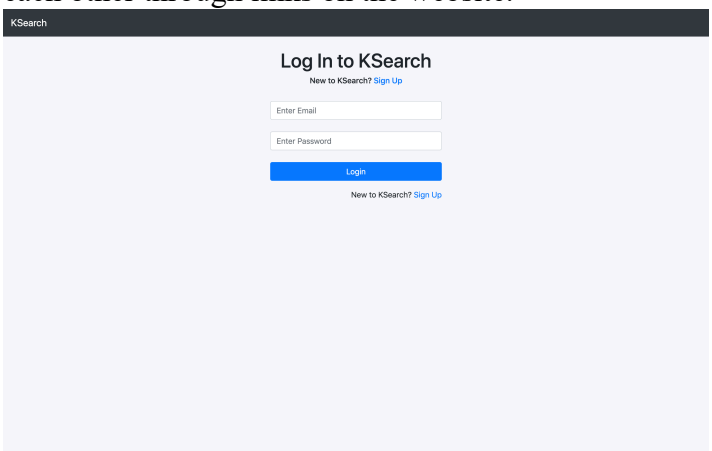
3. Database Visualization

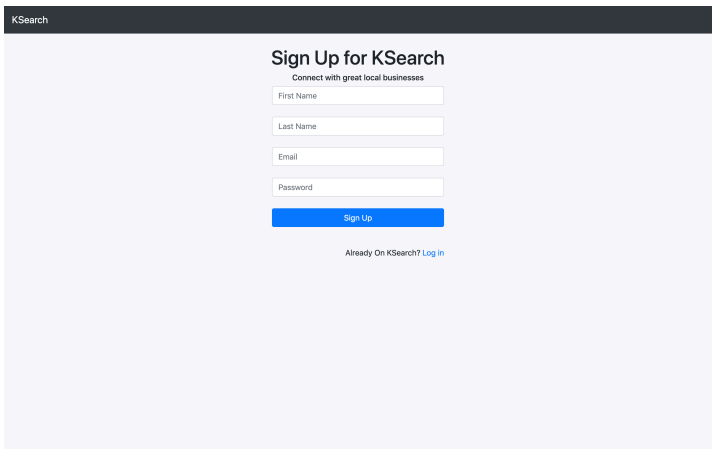


4. Functionality Performed

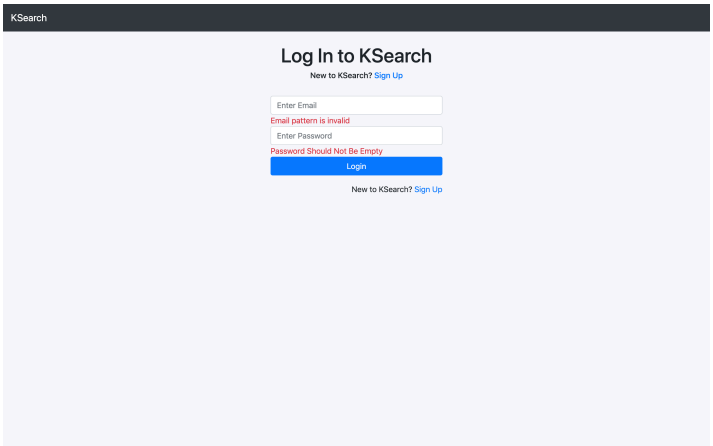
(1) Sign up & Login Page

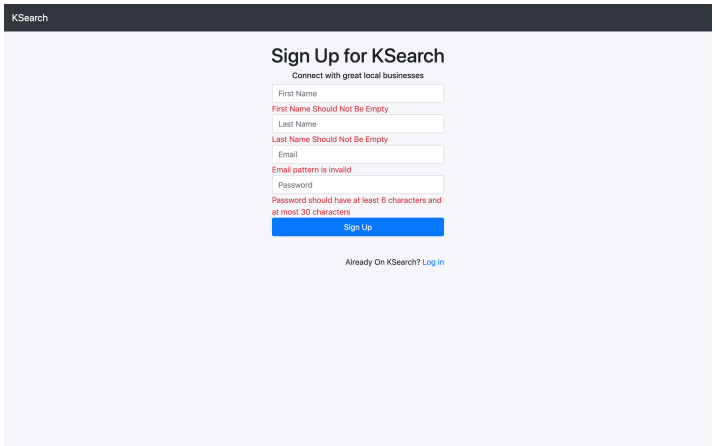
People first visit the website can sign up for KSearch and login into it. The sign up and login page can switch to each other through links on the website.



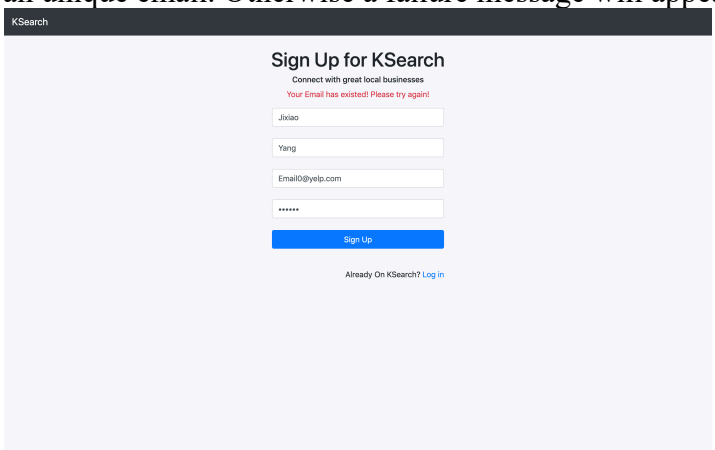


Of course, the input textboxes have validations. The email should satisfy a pattern and the password should have at least 6 characters and at most 30 characters. The first name and last name should not be empty.

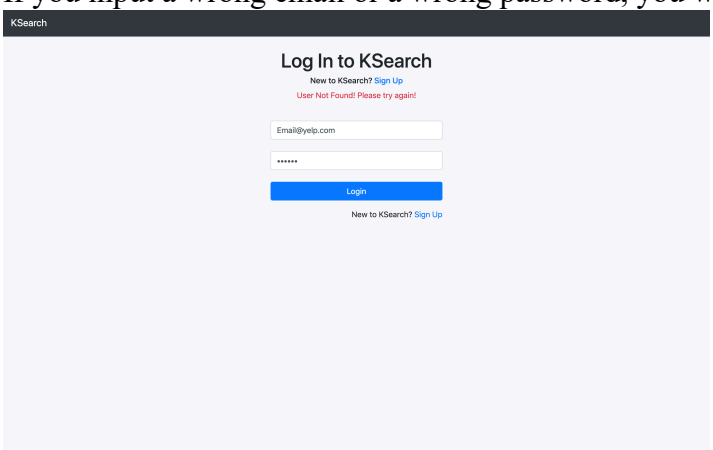


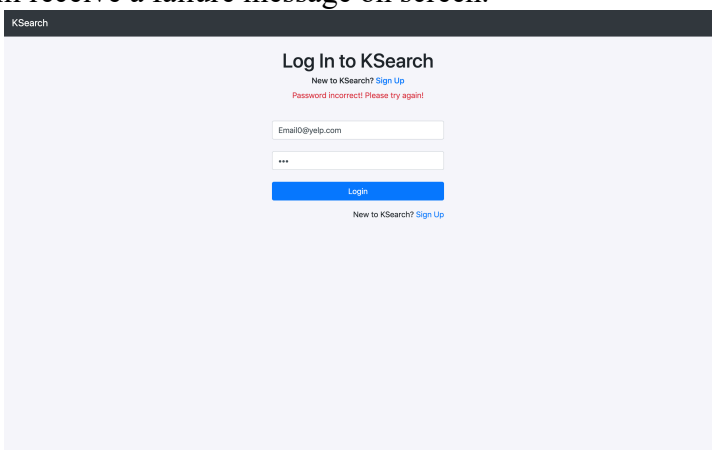


And also, the email must be an unique email. Otherwise a failure message will appear.



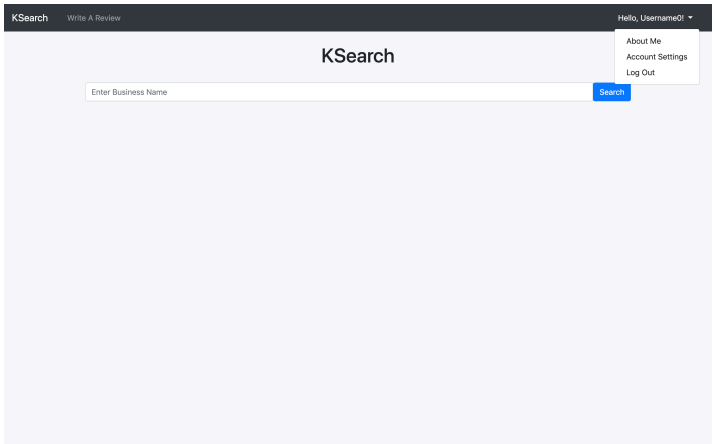
If you input a wrong email or a wrong password, you will receive a failure message on screen.

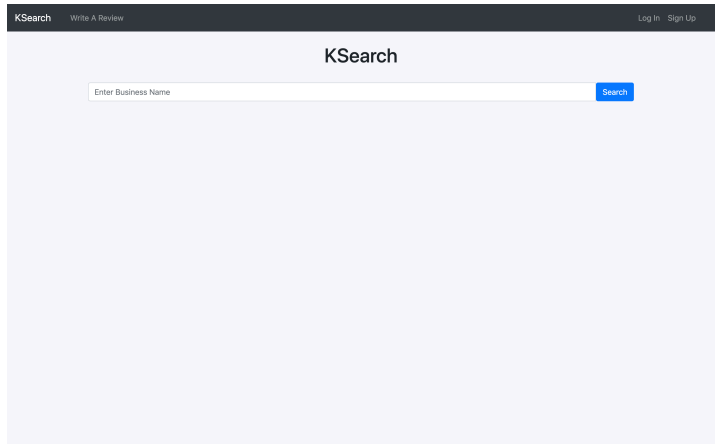




(2) Main Search Page

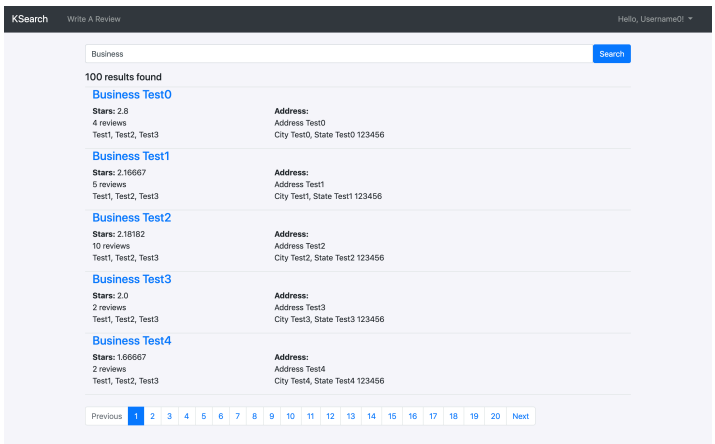
The main page is the first page when people visit KSearch. People can search business on this page weather they are logged in or not.





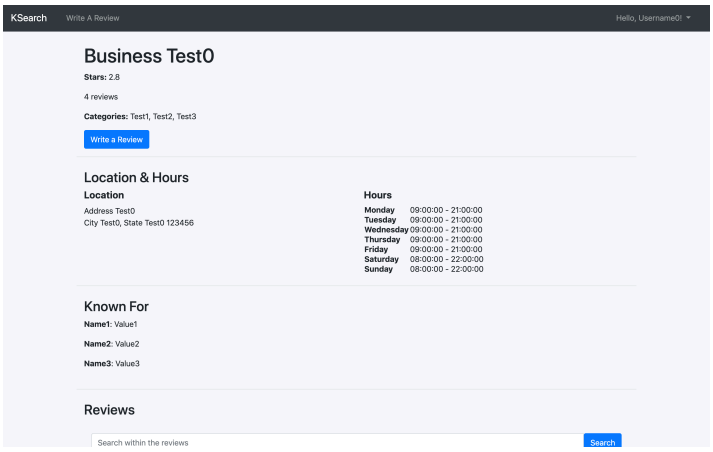
(3) Business Search Result Page

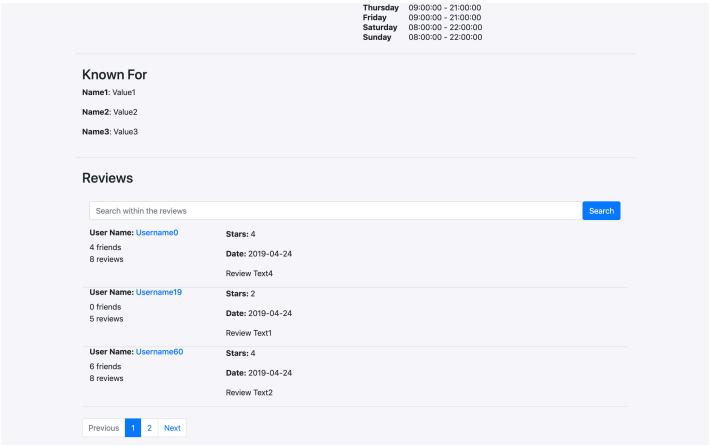
People enter a business name and click on search button to get a business search result page. This page supports pagination, each page shows 5 results. People can check a basic information of businesses and click on a business name to check its details.



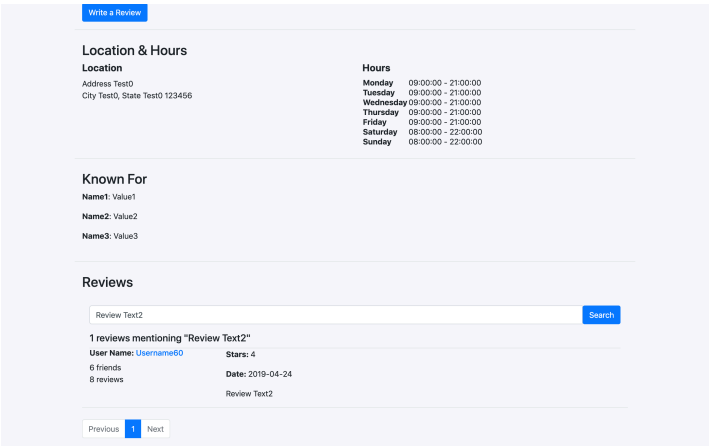
(4) Business Details Page

In the business details page, people can check the explicit information of a business as shown below. People can click on the write a review button to write reviews to the business.



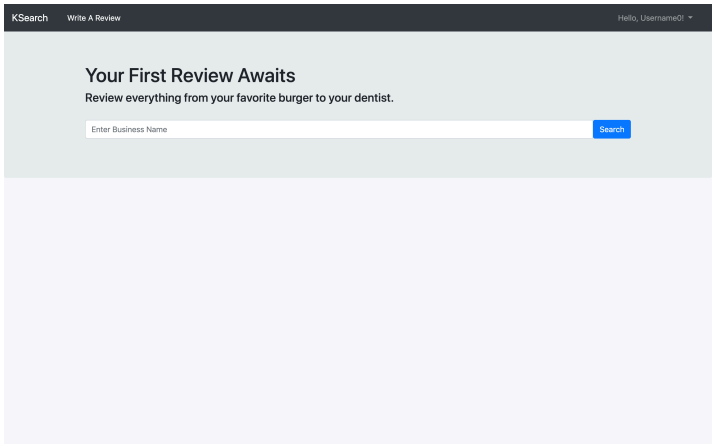


At the review part, people can search related review texts. This part supports pagination. And people can click on the username of to check the user's details.

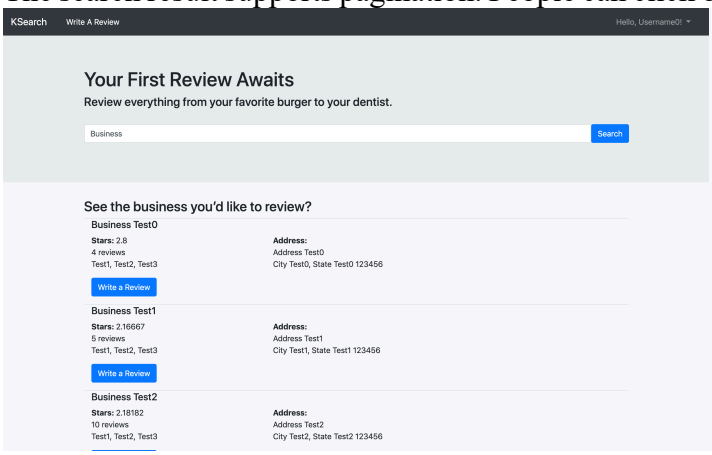


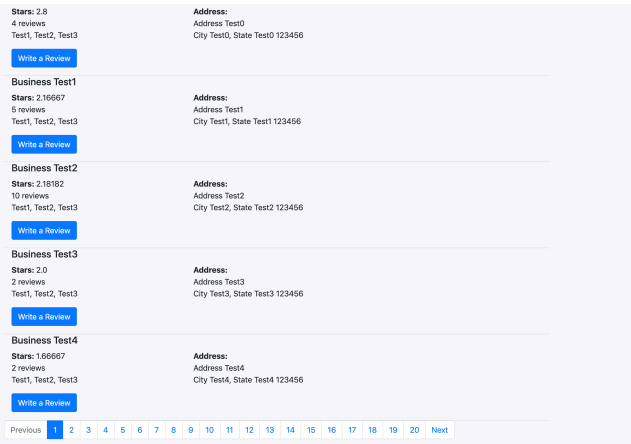
(5) Write A Review Page

In the write a review page, we need to search for a business to write a review.



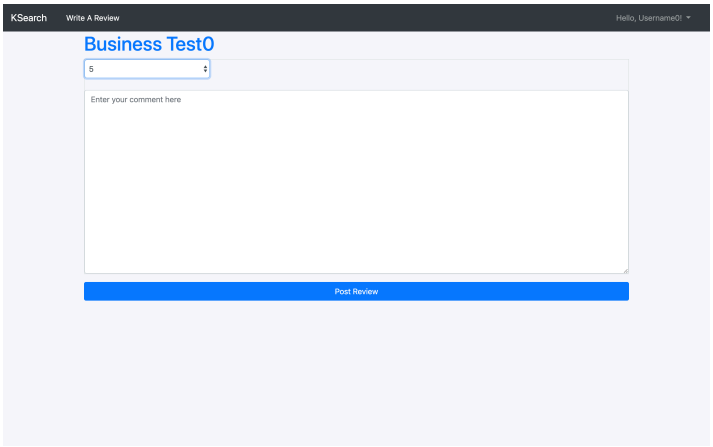
The search result supports pagination. People can click on the write a review button to write reviews to businesses.

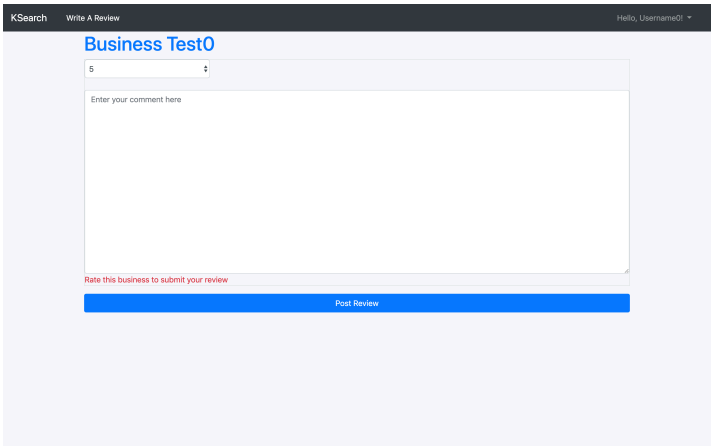




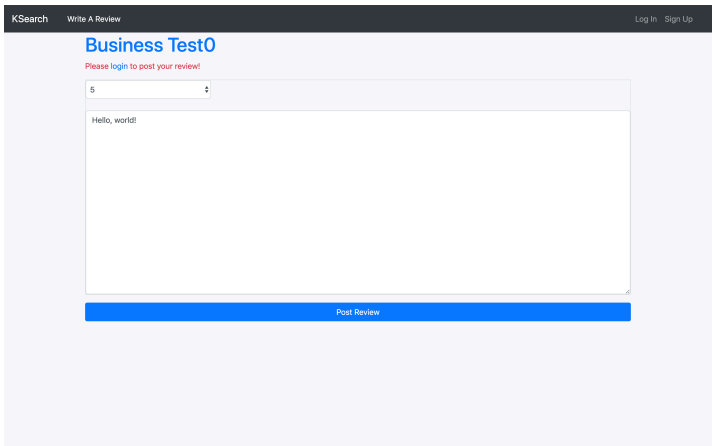
(6) Post Review Page

In the post review page, people can rate a business and write text comments to a business. The rate is among 1, 2, 3, 4, 5 and the text comments cannot be empty. Otherwise you will receive an error message. And you can click on the business name to check details about this business.



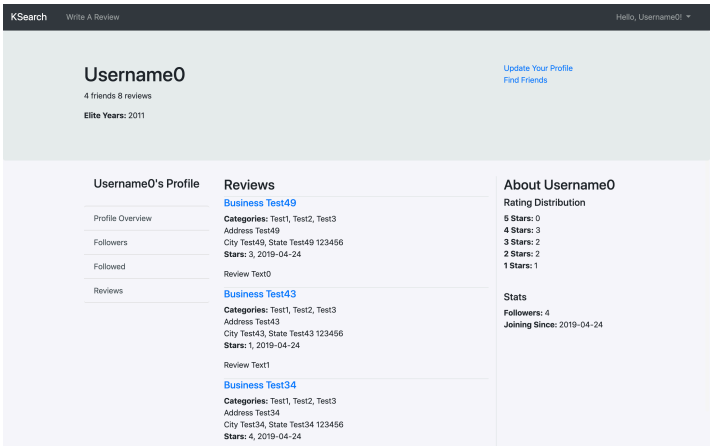


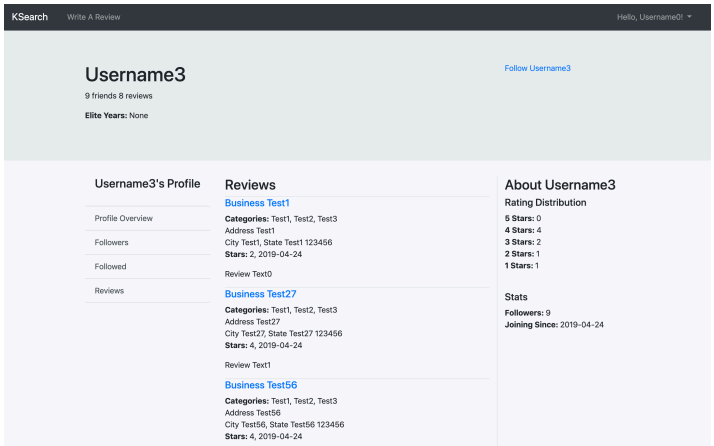
Most importantly, this operation needs people to login. If you don't login to KSearch, you will be asked to login when you click the post review button.



(7) User Details Page – Profile Overview

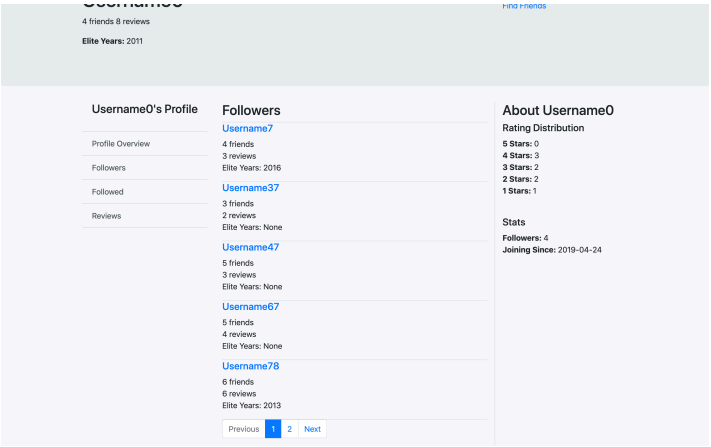
In this page, people can check a user's explicit information. If the user page shows your own account, you can click the update your profile link to update your information and the find friends link to find more users on KSearch. If you are checking other users' information, you can click the follow or unfollow link to follow or unfollow this user. In the meantime, you can click on the business names in this page to check details of the businesses.





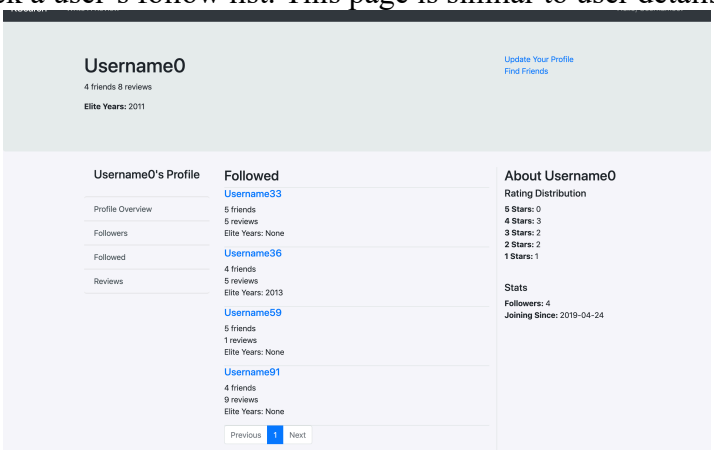
(8) User Details Page – Followers

In this page, you can check a user's follower list. This page supports pagination. Each page shows 5 users and you can click on the user names to check details of the users.



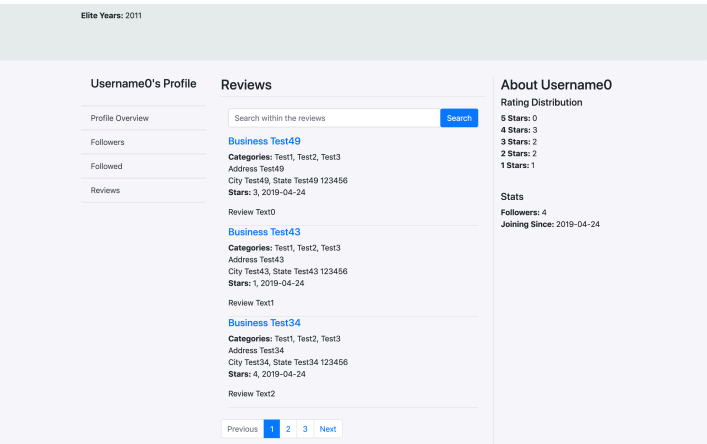
(9) User Details Page – Followed

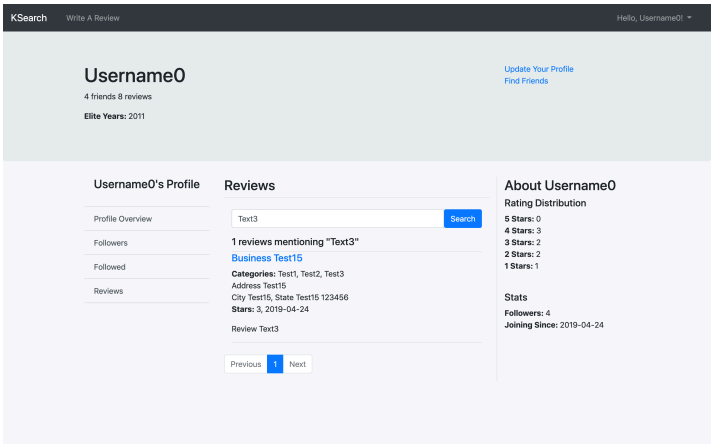
In this page, people can check a user's follow list. This page is similar to user details page – followers.



(10) User Details Page – Reviews

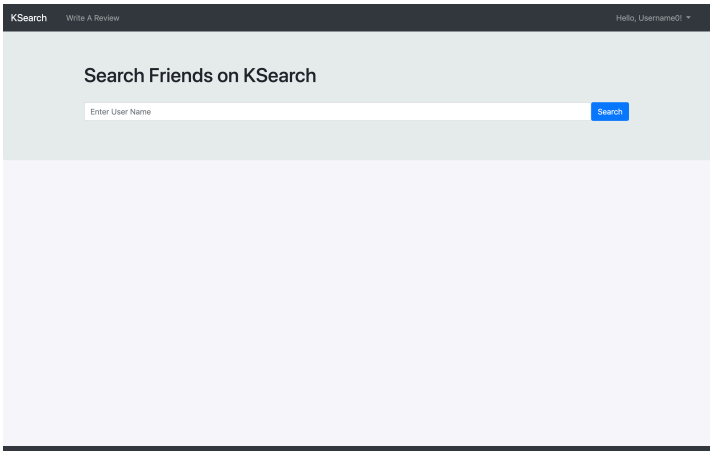
In this page, people can check all of the reviews of a user. People can also search relates review text. This part supports pagination and each page shows 3 results. People can click on the business name to check details of the businesses.

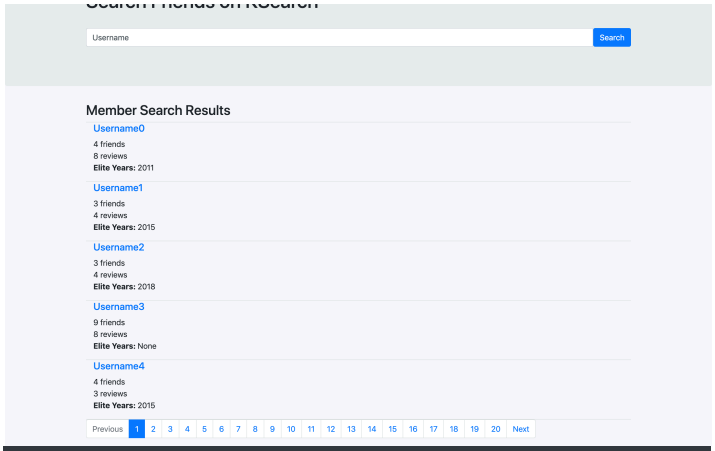


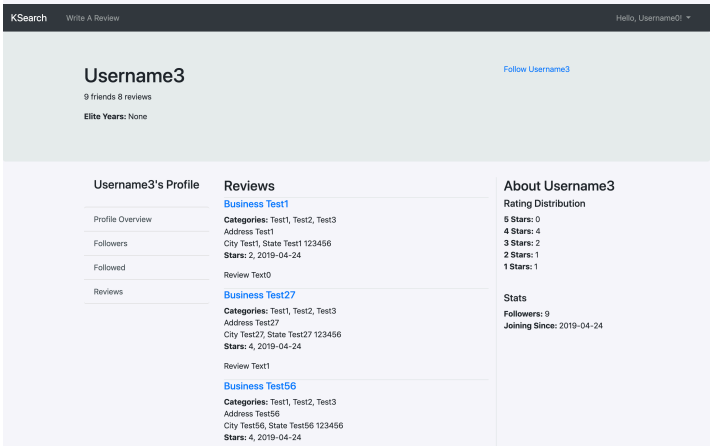


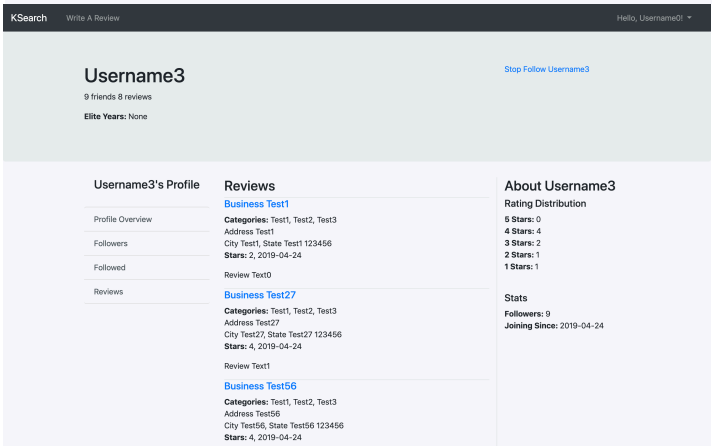
(11) Find Users(Friends) Page

In this page, people can search for other users on KSearch. The search result page supports pagination and each page shows 5 results. People can click on the user names to check details of a user.



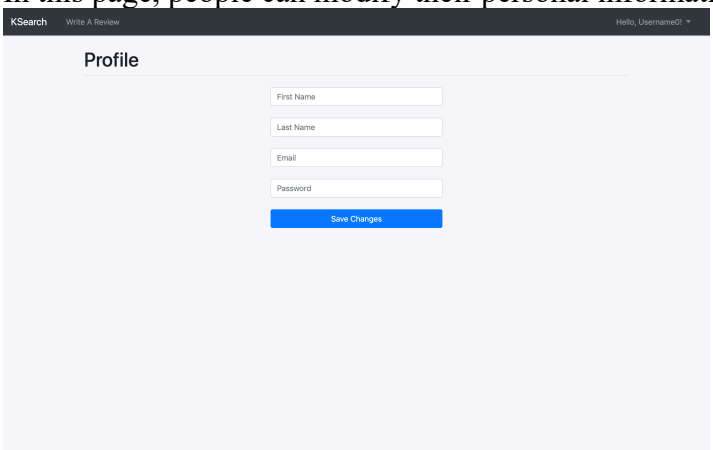


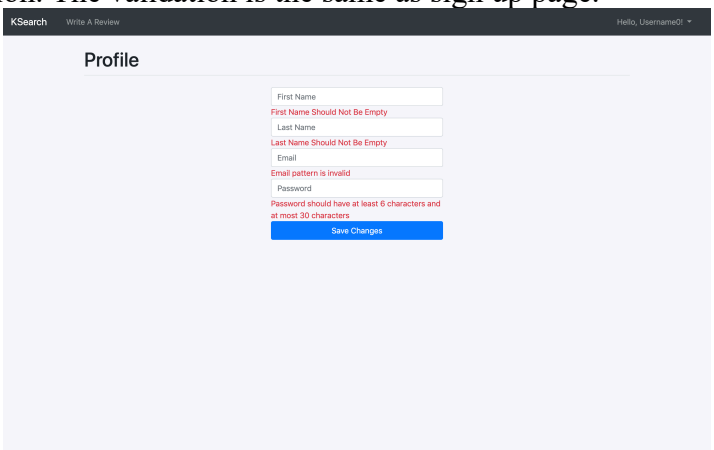


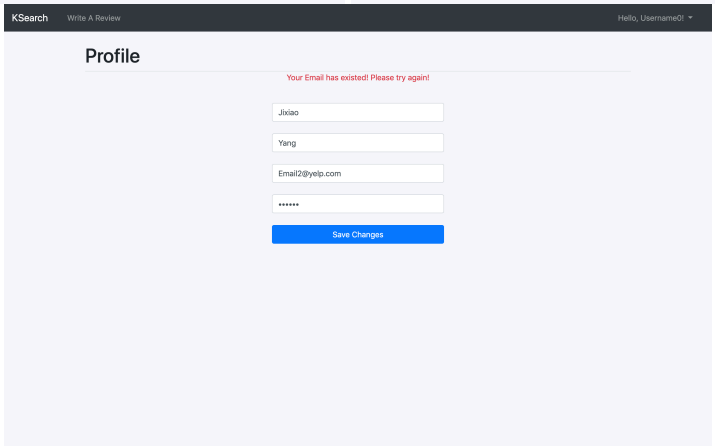


(12) User Account Settings Page

In this page, people can modify their personal information. The validation is the same as sign up page.







APPENDIX

1. Pojos

(1) User

```
package Model.Data;

import Model.Form.*;

import javax.persistence.*;
import java.sql.Date;
import java.util.*;

@Entity
@Table(name = "user")
public class User {
    private int id;
    private String name;
    private String email;
    private String password;
    private Integer reviewCount;
    private Date dateJoined;
    private Set<Integer> followList;
    private Set<Integer> followerList;
    private Set<Integer> elite;
    private Double averageStars;
    private List<Review> reviews;

    public User() {

    }

    public User(UserSignUp userSignUp) {
        this.name = userSignUp.getFirstName() + " " + userSignUp.getLastName();
        this.email = userSignUp.getEmail();
        this.password = userSignUp.getPassword();
        reviewCount = 0;
        dateJoined = new Date(System.currentTimeMillis());
        followList = new HashSet<>();
        followerList = new HashSet<>();
        elite = new HashSet<>();
        averageStars = 0.0;
        reviews = new ArrayList<>();
    }

    public List<Integer> countUserRatingDistribution() {
        List<Integer> result = new ArrayList<Integer>() {{
            add(0);
            add(0);
            add(0);
            add(0);
            add(0);
        }};
        for (Review review : reviews) {
            int stars = review.getStars();
            result.set(stars - 1, result.get(stars - 1) + 1);
        }
        return result;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "user_id")
    public int getId() {
        return id;
    }
```

```java
    public void setId(int id) {
        this.id = id;
    }

    @Column(name = "user_name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Column(name = "email")
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Column(name = "password")
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Column(name = "review_count")
    public Integer getReviewCount() {
        return reviewCount;
    }

    public void setReviewCount(Integer reviewCount) {
        this.reviewCount = reviewCount;
    }

    @Column(name = "yelping_since")
    public Date getDateJoined() {
        return dateJoined;
    }

    public void setDateJoined(Date dateJoined) {
        this.dateJoined = dateJoined;
    }

    @ElementCollection(targetClass = Integer.class)
    @CollectionTable(name = "friends", joinColumns = @JoinColumn(name = "user_id"))
    @Column(name = "friend_id")
    public Set<Integer> getFollowList() {
        return followList;
    }

    public void setFollowList(Set<Integer> followList) {
        this.followList = followList;
    }

    @ElementCollection(targetClass = Integer.class)
    @CollectionTable(name = "friends", joinColumns = @JoinColumn(name = "friend_id"))
    @Column(name = "user_id")
    public Set<Integer> getFollowerList() {
        return followerList;
    }

    public void setFollowerList(Set<Integer> followerList) {
```

```
        this.followerList = followerList;
    }

    @ElementCollection(targetClass = Integer.class)
    @CollectionTable(name = "elite", joinColumns = @JoinColumn(name = "user_id"))
    @Column(name = "elite_year")
    public Set<Integer> getElite() {
        return elite;
    }

    public void setElite(Set<Integer> elite) {
        this.elite = elite;
    }

    @Column(name = "average_stars")
    public Double getAverageStars() {
        return averageStars;
    }

    public void setAverageStars(Double averageStars) {
        this.averageStars = averageStars;
    }

    @OneToMany(mappedBy = "user", targetEntity = Review.class, cascade = CascadeType.ALL,
orphanRemoval = true)
    public List<Review> getReviews() {
        return reviews;
    }

    public void setReviews(List<Review> reviews) {
        this.reviews = reviews;
    }
}
```

(2) Business

```
package Model.Data;

import javax.persistence.*;
import java.util.*;

@Entity
@Table(name = "business")
public class Business {
    private int id;
    private String name;
    private String address;
    private String city;
    private String state;
    private String postalCode;
    private Double latitude;
    private Double longitude;
    private Double stars;
    private Integer reviewCount;
    private Short isOpen;
    private List<Attribute> attributes;
    private List<String> categories;
    private List<Hours> hours;
    private List<Review> reviews;

    public Business() {

    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "business_id")
    public int getId() {
        return id;
    }
```

```java
    public void setId(int id) {
        this.id = id;
    }

    @Column(name = "business_name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Column(name = "address")
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Column(name = "city")
    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    @Column(name = "state")
    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    @Column(name = "postal_code")
    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }

    @Column(name = "latitude")
    public Double getLatitude() {
        return latitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    @Column(name = "longitude")
    public Double getLongitude() {
        return longitude;
    }

    public void setLongitude(Double longitude) {
        this.longitude = longitude;
    }
```

```java
    @Column(name = "stars")
    public Double getStars() {
        return stars;
    }

    public void setStars(Double stars) {
        this.stars = stars;
    }

    @Column(name = "review_count")
    public Integer getReviewCount() {
        return reviewCount;
    }

    public void setReviewCount(Integer reviewCount) {
        this.reviewCount = reviewCount;
    }

    @Column(name = "is_open")
    public Short getIsOpen() {
        return isOpen;
    }

    public void setIsOpen(Short isOpen) {
        this.isOpen = isOpen;
    }

    @OneToMany(mappedBy = "business", targetEntity = Attribute.class, cascade = CascadeType.ALL,
orphanRemoval = true)
    public List<Attribute> getAttributes() {
        return attributes;
    }

    public void setAttributes(List<Attribute> attributes) {
        this.attributes = attributes;
    }

    @ElementCollection(targetClass = String.class)
    @CollectionTable(name = "category", joinColumns = @JoinColumn(name = "business_id"))
    @Column(name = "category")
    public List<String> getCategories() {
        return categories;
    }

    public void setCategories(List<String> categories) {
        this.categories = categories;
    }

    @OneToMany(mappedBy = "business", targetEntity = Hours.class, cascade = CascadeType.ALL,
orphanRemoval = true)
    public List<Hours> getHours() {
        return hours;
    }

    public void setHours(List<Hours> hours) {
        this.hours = hours;
    }

    @OneToMany(mappedBy = "business", targetEntity = Review.class, cascade = CascadeType.ALL,
orphanRemoval = true)
    public List<Review> getReviews() {
        return reviews;
    }

    public void setReviews(List<Review> reviews) {
        this.reviews = reviews;
    }
}
```

(3) Review

```java
package Model.Data;

import Model.Form.StarsComment;

import javax.persistence.*;
import java.sql.*;

@Entity
@Table(name = "review")
public class Review {
    private int reviewId;

    private User user;
    private Business business;

    private Integer stars;
    private Date date;
    private String text;

    public Review() {

    }

    public Review(User user, Business business, StarsComment starsComment) {
        this.user = user;
        this.business = business;
        this.stars = starsComment.getStars();
        this.date = new Date(System.currentTimeMillis());
        this.text = starsComment.getText();
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "review_id")
    public int getReviewId() {
        return reviewId;
    }

    public void setReviewId(int reviewId) {
        this.reviewId = reviewId;
    }

    @ManyToOne(cascade = CascadeType.ALL, optional = false, fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @ManyToOne(cascade = CascadeType.ALL, optional = false, fetch = FetchType.LAZY)
    @JoinColumn(name = "business_id")
    public Business getBusiness() {
        return business;
    }

    public void setBusiness(Business business) {
        this.business = business;
    }

    @Column(name = "stars")
    public Integer getStars() {
        return stars;
    }
```

```java
    public void setStars(Integer stars) {
        this.stars = stars;
    }

    @Column(name = "date")
    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    @Column(name = "text")
    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }
}
```

(4) Hours

```java
package Model.Data;

import javax.persistence.*;
import java.sql.*;

@Entity
@Table(name = "hours")
public class Hours {
    private int id;
    private Business business;
    private String day;
    private Time startTime;
    private Time closeTime;

    public Hours() {

    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @ManyToOne(optional = false)
    @JoinColumn(name = "business_id")
    public Business getBusiness() {
        return business;
    }

    public void setBusiness(Business business) {
        this.business = business;
    }

    @Column(name = "day")
    public String getDay() {
        return day;
    }

    public void setDay(String day) {
```

```
        this.day = day;
    }

    @Column(name = "start_time")
    public Time getStartTime() {
        return startTime;
    }

    public void setStartTime(Time startTime) {
        this.startTime = startTime;
    }

    @Column(name = "close_time")
    public Time getCloseTime() {
        return closeTime;
    }

    public void setCloseTime(Time closeTime) {
        this.closeTime = closeTime;
    }
}
```

(5) Attribute

```
package Model.Data;

import javax.persistence.*;

@Entity
@Table(name = "attributes")
public class Attribute {
    private int id;
    private Business business;
    private String name;
    private String value;

    public Attribute() {

    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @ManyToOne(optional = false)
    @JoinColumn(name = "business_id")
    public Business getBusiness() {
        return business;
    }

    public void setBusiness(Business business) {
        this.business = business;
    }

    @Column(name = "name")
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    @Column(name = "value")
    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }
}
```

2. DAOs

(1) DAO

```
package DAO;

import org.hibernate.*;
import org.hibernate.cfg.*;

public class DAO {
    private SessionFactory sessionFactory;
    private static Session session;

    protected SessionFactory setUp() {
        Configuration configuration = new Configuration();
        configuration.configure("hibernate.cfg.xml");
        try {
            sessionFactory = configuration.buildSessionFactory();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return sessionFactory;
    }

    public Session getSession() {
        if (session == null) {
            session = setUp().openSession();
        }
        return session;
    }

    public void beginTransaction() {
        if (!isTransactionActive()) {
            getSession().beginTransaction();
        }
    }

    private boolean isTransactionActive() {
        return getSession().getTransaction().isActive();
    }

    public void commit() {
        getSession().getTransaction().commit();
    }

    public void close() {
        getSession().close();
    }

    public void rollbackTransaction() {
        getSession().getTransaction().rollback();
    }
}
```

(2) UserDAO

```
package DAO;

import Model.Data.*;
import org.hibernate.*;
import org.hibernate.criterion.*;
```

```java
import java.util.*;

@SuppressWarnings("deprecation")
public class UserDAO extends DAO {

    public User getUserById(int id) {
        User user;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("from User where id=:id");
            query.setInteger("id", id);
            query.setMaxResults(1);
            user = (User) query.uniqueResult();

        }  catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return user;
    }

    public User getUserByEmail(String email) {
        User user;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("from User where email=:email");
            query.setString("email", email);
            query.setMaxResults(1);
            user = (User) query.uniqueResult();

        }  catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return user;
    }

    public int getUserCount(String keyword) {
        Long result;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("select count(*) from User where name
like :keyword");
            query.setString("keyword", "%" + keyword + "%");
            result = (Long)query.uniqueResult();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return -1;
        }
        return result.intValue();
    }

    public List<User> getUsers(String keyword, int page, int eachPageCount) {
        List<User> result;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("from User where name like :keyword");
            query.setString("keyword", "%" + keyword + "%");
            query.setFirstResult((page - 1) * eachPageCount);
            query.setMaxResults(eachPageCount);
```

```
            result = query.list();

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return result;
    }

    public boolean addUser(User User) {
        boolean result;
        try {
            beginTransaction();
            Session session = getSession();
            session.save(User);
            commit();
            result = true;

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return false;
        }
        return result;
    }

    public boolean findExistedEmail(String email) {
        boolean result = true;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("select count(*) from User where email=:email");
            query.setString("email", email);
            long count = (long)query.uniqueResult();
            if (count == 0) {
                result = false;
            }

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return true;
        }
        return result;
    }

    public boolean updateUser(User user) {
        boolean result = false;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("update User set name=:name, password=:password,
email=:email where id=:id");
            query.setString("name", user.getName());
            query.setString("password", user.getPassword());
            query.setString("email", user.getEmail());
            query.setInteger("id", user.getId());
            int modifyResult = query.executeUpdate();
            if (modifyResult > 0) {
                result = true;
            }
            commit();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return false;
        }
```

```java
        return result;
    }

    public boolean followUser(int userId, int friendId) {
        boolean result;
        try {
            beginTransaction();
            Session session = getSession();
            User user = session.get(User.class, userId);
            user.getFollowList().add(friendId);
            session.update(user);
            commit();
            result = true;
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return false;
        }
        return result;
    }

    public boolean unfollowUser(int userId, int friendId) {
        boolean result;
        try {
            beginTransaction();
            Session session = getSession();
            User user = session.get(User.class, userId);
            user.getFollowList().remove(friendId);
            session.update(user);
            commit();
            result = true;
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return false;
        }
        return result;
    }

    public List<User> getFollowers(int userId, int page, int eachPageCount) {
        List<User> result = new ArrayList<>();
        try {
            beginTransaction();
            Session session = getSession();
            User user = session.get(User.class, userId);
            Set<Integer> followerId = user.getFollowerList();
            if (followerId.size() == 0) {
                return result;
            }
            Criteria criteria = session.createCriteria(User.class);
            Disjunction disjunction = Restrictions.disjunction();
            for (Integer id : followerId) {
                disjunction.add(Restrictions.eq("id", id));
            }
            criteria.add(disjunction);
            criteria.setFirstResult((page - 1) * eachPageCount);
            criteria.setMaxResults(eachPageCount);
            result = criteria.list();

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return result;
    }

    public List<User> getFollows(int userId, int page, int eachPageCount) {
```

```
        List<User> result = new ArrayList<>();
        try {
            beginTransaction();
            Session session = getSession();
            User user = session.get(User.class, userId);
            Set<Integer> followId = user.getFollowList();
            if (followId.size() == 0) {
                return result;
            }
            Criteria criteria = session.createCriteria(User.class);
            Disjunction disjunction = Restrictions.disjunction();
            for (Integer id : followId) {
                disjunction.add(Restrictions.eq("id", id));
            }
            criteria.add(disjunction);
            criteria.setFirstResult((page - 1) * eachPageCount);
            criteria.setMaxResults(eachPageCount);
            result = criteria.list();

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return result;
    }
}
```

(3) BusinessDAO

```
package DAO;

import java.util.*;

import Model.Data.*;
import org.hibernate.*;

@SuppressWarnings("deprecation")
public class BusinessDAO extends DAO {

    public Business getBusiness(int id) {
        Business business;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("from Business where id = :id");
            query.setInteger("id", id);
            business = (Business)query.uniqueResult();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return business;
    }

    public int getBusinessCount(String keyword) {
        Long result;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("select count(*) from Business where name
like :keyword");
            query.setString("keyword", "%" + keyword + "%");
            result = (Long)query.uniqueResult();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return -1;
        }
```

```
        return result.intValue();
    }

    public List<Business> getBusinesses(String keyword, int page, int eachPageCount) {
        List<Business> result;
        try {
            beginTransaction();
            Session session = getSession();
            Query query = session.createQuery("from Business where name like :keyword");
            query.setString("keyword", "%" + keyword + "%");
            query.setFirstResult((page - 1) * eachPageCount);
            query.setMaxResults(eachPageCount);
            result = query.list();

        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return result;
    }

}
```

(4) ReviewDAO

```
package DAO;

import java.util.*;

import Model.Data.*;
import Model.Form.*;
import org.hibernate.*;
import org.hibernate.criterion.*;

@SuppressWarnings("deprecation")
public class ReviewDAO extends DAO {

    public int getReviewCountByBusinessId(int id, String keyword) {
        Long result;
        try {
            beginTransaction();
            Session session = getSession();
            Criteria criteria = session.createCriteria(Review.class);
            Criteria businessCriteria = criteria.createCriteria("business");
            businessCriteria.add(Restrictions.eq("id", id));
            if (keyword != null && !keyword.equals("")) {
                criteria.add(Restrictions.like("text", keyword, MatchMode.ANYWHERE));
            }
            criteria.setProjection(Projections.rowCount());
            result = (Long)criteria.uniqueResult();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return -1;
        }
        return result.intValue();
    }

    public List<Review> getReviewsByBusinessId(int id, String keyword, int page, int eachPageCount)
{
        List<Review> result;
        try {
            beginTransaction();
            Session session = getSession();
            Criteria criteria = session.createCriteria(Review.class);
            Criteria businessCriteria = criteria.createCriteria("business");
            businessCriteria.add(Restrictions.eq("id", id));
            if (keyword != null && !keyword.equals("")) {
                criteria.add(Restrictions.like("text", keyword, MatchMode.ANYWHERE));
```

```java
        }
        criteria.setFirstResult((page - 1) * eachPageCount);
        criteria.setMaxResults(eachPageCount);
        result = criteria.list();
    } catch (HibernateException e) {
        rollbackTransaction();
        e.printStackTrace();
        return null;
    }
    return result;
}

public List<Review> getReviewsByUserLimited(int userId, int limitCount) {
    List<Review> result;
    try {
        beginTransaction();
        Session session = getSession();
        Criteria criteria = session.createCriteria(Review.class);
        Criteria businessCriteria = criteria.createCriteria("user");
        businessCriteria.add(Restrictions.eq("id", userId));
        criteria.setMaxResults(limitCount);
        result = criteria.list();
    } catch (HibernateException e) {
        rollbackTransaction();
        e.printStackTrace();
        return null;
    }
    return result;
}

public boolean addReview(int userId, int businessId, StarsComment starsComment) {
    try {
        beginTransaction();
        Session session = getSession();
        User user = session.get(User.class, userId);
        Business business = session.get(Business.class, businessId);

        Review review = new Review(user, business, starsComment);

        List<Review> userReviews = user.getReviews();
        userReviews.add(review);
        user.setReviewCount(userReviews.size());
        double tmp = 0.0;
        for (Review userReview : userReviews) {
            tmp += userReview.getStars();
        }
        user.setAverageStars(tmp / user.getReviewCount());

        List<Review> businessReviews = business.getReviews();
        businessReviews.add(review);
        business.setReviewCount(businessReviews.size());
        tmp = 0.0;
        for (Review businessReview : businessReviews) {
            tmp += businessReview.getStars();
        }
        business.setStars(tmp / user.getReviewCount());

        session.save(review);
        session.update(user);
        session.update(business);
        commit();
    } catch (HibernateException e) {
        rollbackTransaction();
        e.printStackTrace();
        return false;
    }
    return true;
}
```

```java
    public int getReviewCountByUserId(int id, String keyword) {
        Long result;
        try {
            beginTransaction();
            Session session = getSession();
            Criteria criteria = session.createCriteria(Review.class);
            Criteria businessCriteria = criteria.createCriteria("user");
            businessCriteria.add(Restrictions.eq("id", id));
            if (keyword != null && !keyword.equals("")) {
                criteria.add(Restrictions.like("text", keyword, MatchMode.ANYWHERE));
            }
            criteria.setProjection(Projections.rowCount());
            result = (Long)criteria.uniqueResult();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return -1;
        }
        return result.intValue();
    }

    public List<Review> getReviewsByUserId(int id, String keyword, int page, int eachPageCount) {
        List<Review> result;
        try {
            beginTransaction();
            Session session = getSession();
            Criteria criteria = session.createCriteria(Review.class);
            Criteria businessCriteria = criteria.createCriteria("user");
            businessCriteria.add(Restrictions.eq("id", id));
            if (keyword != null && !keyword.equals("")) {
                criteria.add(Restrictions.like("text", keyword, MatchMode.ANYWHERE));
            }
            criteria.setFirstResult((page - 1) * eachPageCount);
            criteria.setMaxResults(eachPageCount);
            result = criteria.list();
        } catch (HibernateException e) {
            rollbackTransaction();
            e.printStackTrace();
            return null;
        }
        return result;
    }
}
```

3. Controllers

(1) MainController

```java
package Controller;

import javax.servlet.http.*;

import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class MainController {

    @RequestMapping(path = "/main", method = RequestMethod.GET)
    public String mainHandler(HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("loggedInUser") == null) {
            session.setAttribute("loggedInUser", null);
        }
        return "main";
    }

    @RequestMapping(path = "/logout", method = RequestMethod.GET)
    public String dealWithLogout(HttpServletRequest request) {
```

```
        HttpSession session = request.getSession();
        session.setAttribute("loggedInUser", null);
        return "main";
    }

}
```

(2) SearchBusinessController

```
package Controller;

import DAO.*;
import Model.Data.*;
import java.util.*;
import javax.servlet.http.*;


import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class SearchBusinessController {

    @RequestMapping(path = "/search", method = RequestMethod.POST)
    public String searchBusiness(HttpServletRequest request, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        String keyword = request.getParameter("keyword");
        if (keyword == null) {
            keyword = "";
        }
        int resultCount = businessDAO.getBusinessCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Business> resultList = businessDAO.getBusinesses(keyword, 1, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);
        return "search";
    }

    @RequestMapping(path = "/search", method = RequestMethod.GET)
    public String searchBusiness(@RequestParam("keyword") String keyword, @RequestParam("page") int
page, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        int resultCount = businessDAO.getBusinessCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Business> resultList = businessDAO.getBusinesses(keyword, page, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);
        return "search";
    }


}
```

(3) SignUpController

```
package Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class SignUpController {

    @RequestMapping(path = "/signup", method = RequestMethod.GET)
    public String getUserSignUp(ModelMap model, UserSignUp user) {
        model.addAttribute("userSignUp", user);
        return "signup";
    }

    @RequestMapping(path = "/signup", method = RequestMethod.POST)
    public String dealWithSignUp(@Validated @ModelAttribute("userSignUp") UserSignUp userSignUp,
BindingResult bindingResult, ModelMap model) {
        if (bindingResult.hasErrors()) {
            return "signup";
        }

        UserDAO userDAO = new UserDAO();
        boolean existed = userDAO.findExistedEmail(userSignUp.getEmail());
        System.out.println("Email existed? " + existed);
        if (existed) {
            model.addAttribute("failure", "Your Email has existed! Please try again!");
            return "signup";
        }

        User user = new User(userSignUp);
        boolean result = userDAO.addUser(user);
        System.out.println("Add User Result: " + result);
        if (!result) {
            model.addAttribute("failure", "Sign Up Failed! Please try again!");
            return "signup";
        }

        return "redirect:/login";
    }
}
```

(4) LoginController

```
package Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import javax.servlet.http.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class LoginController {

    @RequestMapping(path = "/login", method = RequestMethod.GET)
    public String getUserLogin(ModelMap model, UserLogin user) {
        model.addAttribute("userLogin", user);
        return "login";
```

```
    }

    @RequestMapping(path = "/login", method = RequestMethod.POST)
    public String dealWithLogin(@Validated @ModelAttribute("userLogin") UserLogin userLogin,
BindingResult result, ModelMap model, HttpServletRequest request) {
        if (result.hasErrors()) {
            return "login";
        }

        UserDAO userDAO = new UserDAO();
        User user = userDAO.getUserByEmail(userLogin.getEmail());
        if (user == null) {
            model.addAttribute("failure", "User Not Found! Please try again!");
            return "login";
        } else if (!user.getPassword().equals(userLogin.getPassword())) {
            model.addAttribute("failure", "Password incorrect! Please try again!");
            return "login";
        }

        HttpSession session = request.getSession();
        session.setAttribute("loggedInUser", user);

        return "redirect:/main";
    }
}
```

(5) BusinessController

```
package Controller;

import DAO.*;
import Model.Data.*;
import java.util.*;
import javax.servlet.http.*;


import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/business")
public class BusinessController {

    @RequestMapping(path = "/{businessId:[\\d]+}", method = {RequestMethod.POST,
RequestMethod.GET})
    public String showBusiness(@PathVariable("businessId") int businessId,
                               HttpServletRequest request, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        ReviewDAO reviewDAO = new ReviewDAO();
        Business business = businessDAO.getBusiness(businessId);

        String keyword = request.getParameter("keyword");
        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = business.getReviews().size();
        } else {
            resultCount = reviewDAO.getReviewCountByBusinessId(businessId, keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviews = reviewDAO.getReviewsByBusinessId(businessId, keyword, 1,
eachPageCount);
        model.addAttribute("business", business);
        model.addAttribute("reviews", reviews);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
```

```
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        return "business";
    }

    @RequestMapping(path = "/{businessId:[\\d]+}/search_review", method = RequestMethod.GET)
    public String showBusiness(@PathVariable("businessId") int businessId, @RequestParam("keyword")
String keyword,
                               @RequestParam("page") int page, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        ReviewDAO reviewDAO = new ReviewDAO();
        Business business = businessDAO.getBusiness(businessId);

        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = business.getReviews().size();
        } else {
            resultCount = reviewDAO.getReviewCountByBusinessId(businessId, keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviews = reviewDAO.getReviewsByBusinessId(businessId, keyword, page,
eachPageCount);
        model.addAttribute("business", business);
        model.addAttribute("reviews", reviews);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        return "business";
    }

}
```

(6) WriteReviewController

```
package Controller;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import java.util.*;
import javax.servlet.http.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/writeareview")
public class WriteReviewController {

    @RequestMapping(path = "", method = RequestMethod.GET)
    public String showView() {
        return "writeareview";
    }

    @RequestMapping(path = "/search", method = RequestMethod.POST)
    public String searchBusiness(HttpServletRequest request, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        String keyword = request.getParameter("keyword");
        if (keyword == null) {
            keyword = "";
        }
        int resultCount = businessDAO.getBusinessCount(keyword);
```

```
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Business> resultList = businessDAO.getBusinesses(keyword, 1, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);

        return "writeareview";
    }

    @RequestMapping(path = "/search", method = RequestMethod.GET)
    public String searchBusiness(@RequestParam("keyword") String keyword, @RequestParam("page") int
page, ModelMap model) {
        BusinessDAO businessDAO = new BusinessDAO();
        if (keyword == null) {
            keyword = "";
        }
        int resultCount = businessDAO.getBusinessCount(keyword);

        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Business> resultList = businessDAO.getBusinesses(keyword, page, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);

        return "writeareview";
    }

    @RequestMapping(path = "/business/{businessId:[\\d]+}", method = RequestMethod.GET)
    public String writeReview(@PathVariable("businessId") int businessId, ModelMap model,
StarsComment review) {
        BusinessDAO businessDAO = new BusinessDAO();
        Business business = businessDAO.getBusiness(businessId);
        model.addAttribute("business", business);
        model.addAttribute("starsComment", review);
        return "rate";
    }

    @RequestMapping(path = "/business/{businessId:[\\d]+}", method = RequestMethod.POST)
    public String submitReview(@PathVariable("businessId") int businessId, ModelMap model,
HttpServletRequest request,
                               @Validated @ModelAttribute("starsComment") StarsComment
starsComment, BindingResult result) {
        BusinessDAO businessDAO = new BusinessDAO();
        Business business = businessDAO.getBusiness(businessId);
        model.addAttribute("business", business);

        if (result.hasErrors()) {
            return "rate";
        }

        HttpSession session = request.getSession();
        if (session.getAttribute("loggedInUser") == null) {
            model.addAttribute("failure", true);
            return "rate";
```

```
        }

        User user = (User) session.getAttribute("loggedInUser");
        ReviewDAO reviewDAO = new ReviewDAO();
        if (!reviewDAO.addReview(user.getId(), businessId, starsComment)) {
            model.addAttribute("postFailure", true);
            return "rate";
        }

        return "redirect:/business/" + business.getId() + "?page=1";
    }

}
```

(7) UserDetailsController

```
package Controller;

import java.util.*;
import javax.servlet.http.*;

import DAO.*;
import Model.Data.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/user_details")
public class UserDetailsController {

    @RequestMapping(path = "", method = RequestMethod.GET)
    public String getUserDetails(@RequestParam("user_id") int userId,  HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        if (user == null || user.getId() != userId) {
            UserDAO userDAO = new UserDAO();
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        List<Review> reviews = reviewDAO.getReviewsByUserLimited(userId, 3);
        List<Integer> ratingDistribution = user.countUserRatingDistribution();

        model.addAttribute("user", user);
        model.addAttribute("showReviews", reviews);
        model.addAttribute("ratingDistribution", ratingDistribution);

        return "userdetails";
    }

    @RequestMapping(path = "/followers", method = RequestMethod.GET)
    public String getUserFollowers(@RequestParam("user_id") int userId, HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowerList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followerList = userDAO.getFollowers(user.getId(), 1, eachPageCount);
```

```
        model.addAttribute("user", user);
        model.addAttribute("followerList", followerList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/followers_pagination", method = RequestMethod.GET)
    public String getUserFollowers(@RequestParam("user_id") int userId, @RequestParam("page") int
page,
                                   HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowerList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followerList = userDAO.getFollowers(user.getId(), page, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followerList", followerList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/follows", method = RequestMethod.GET)
    public String getUserFollows(@RequestParam("user_id") int userId, HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followList = userDAO.getFollows(user.getId(), 1, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followList", followList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
```

```
    }

    @RequestMapping(path = "/follows_pagination", method = RequestMethod.GET)
    public String getUserFollows(@RequestParam("user_id") int userId, @RequestParam("page") int
page,
                                 HttpServletRequest request, ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        int resultCount = user.getFollowList().size();
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }

        List<User> followList = userDAO.getFollows(user.getId(), page, eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("followList", followList);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
    }

    @RequestMapping(path = "/reviews", method = {RequestMethod.GET, RequestMethod.POST})
    public String getUserReviews(@RequestParam("user_id") int userId, HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        String keyword = request.getParameter("keyword");
        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = user.getReviews().size();
        } else {
            resultCount = reviewDAO.getReviewCountByUserId(userId, keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviewList = reviewDAO.getReviewsByUserId(userId, keyword, 1, eachPageCount);


        model.addAttribute("user", user);
        model.addAttribute("reviewList", reviewList);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
    }
```

```
    @RequestMapping(path = "/reviews_search", method = RequestMethod.GET)
    public String getUserReviews(@RequestParam("user_id") int userId, @RequestParam("keyword")
String keyword,
                                 @RequestParam("page") int page, HttpServletRequest request,
ModelMap model) {
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("loggedInUser");
        UserDAO userDAO = new UserDAO();
        if (user == null || user.getId() != userId) {
            user = userDAO.getUserById(userId);
        }

        ReviewDAO reviewDAO = new ReviewDAO();
        int resultCount;
        if (keyword == null || keyword.equals("")) {
            resultCount = user.getReviews().size();
        } else {
            resultCount = reviewDAO.getReviewCountByUserId(userId, keyword);
        }
        int eachPageCount = 3;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<Review> reviewList = reviewDAO.getReviewsByUserId(userId, keyword, page,
eachPageCount);

        model.addAttribute("user", user);
        model.addAttribute("reviewList", reviewList);
        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("ratingDistribution", user.countUserRatingDistribution());

        return "userdetails";
    }


}
```

(8) UserProfileController

```
package Controller;

import java.util.*;
import javax.servlet.http.*;

import DAO.*;
import Model.Data.*;
import Model.Form.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.validation.*;
import org.springframework.validation.annotation.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class UserProfileController {

    @RequestMapping(path = "/profile", method = RequestMethod.GET)
    public String modifyProfile(ModelMap model, UserSignUp user) {
        model.addAttribute("userSignUp", user);
        return "profile";
    }

    @RequestMapping(path = "/profile", method = RequestMethod.POST)
    public String dealWithSignUp(@Validated @ModelAttribute("userSignUp") UserSignUp userSignUp,
BindingResult bindingResult, HttpServletRequest request, ModelMap model) {
```

```java
        if (bindingResult.hasErrors()) {
            return "profile";
        }

        HttpSession session = request.getSession();
        User loggedInUser = (User) session.getAttribute("loggedInUser");

        UserDAO userDAO = new UserDAO();
        if (!loggedInUser.getEmail().equals(userSignUp.getEmail())) {
            boolean existed = userDAO.findExistedEmail(userSignUp.getEmail());
            System.out.println("Email existed? " + existed);
            if (existed) {
                model.addAttribute("failure", "Your Email has existed! Please try again!");
                return "profile";
            }
        }

        loggedInUser.setName(userSignUp.getFirstName() + " " + userSignUp.getLastName());
        loggedInUser.setPassword(userSignUp.getPassword());
        loggedInUser.setEmail(userSignUp.getEmail());

        boolean result = userDAO.updateUser(loggedInUser);
        System.out.println("Modify User Result: " + result);
        if (!result) {
            model.addAttribute("failure", "Modify Profile Failed! Please try again!");
            return "profile";
        }
        session.setAttribute("loggedInUser", loggedInUser);

        return "redirect:/user_details?user_id=" + loggedInUser.getId();
    }
}
```

(9) FindFriendsController

```java
package Controller;

import java.util.*;
import javax.servlet.http.*;

import DAO.*;
import Model.Data.*;
import org.springframework.stereotype.*;
import org.springframework.ui.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class FindFriendsController {

    @RequestMapping(path = "/search_friends", method = RequestMethod.GET)
    public String findFriends() {
        return "findfriends";
    }

    @RequestMapping(path = "/find_friends", method = RequestMethod.POST)
    public String searchUser(HttpServletRequest request, ModelMap model) {
        UserDAO userDAO = new UserDAO();
        String keyword = request.getParameter("keyword");
        if (keyword == null) {
            keyword = "";
        }
        int resultCount = userDAO.getUserCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<User> resultList = userDAO.getUsers(keyword, 1, eachPageCount);

        model.addAttribute("keyword", keyword);
```

```
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", 1);
        model.addAttribute("searchResult", resultList);

        return "findfriends";
    }

    @RequestMapping(path = "/find_friends", method = RequestMethod.GET)
    public String searchUser(@RequestParam("keyword") String keyword, @RequestParam("page") int
page, ModelMap model) {
        UserDAO userDAO = new UserDAO();

        int resultCount = userDAO.getUserCount(keyword);
        int eachPageCount = 5;
        int pageCount = resultCount / eachPageCount;
        if (resultCount % eachPageCount != 0) {
            pageCount++;
        }
        List<User> resultList = userDAO.getUsers(keyword, page, eachPageCount);

        model.addAttribute("keyword", keyword);
        model.addAttribute("resultCount", resultCount);
        model.addAttribute("pageCount", pageCount);
        model.addAttribute("currentPage", page);
        model.addAttribute("searchResult", resultList);

        return "findfriends";
    }

    @RequestMapping(path = "/follow", method = RequestMethod.GET)
    public String followUser(@RequestParam("user_id") int userId, HttpServletRequest request,
ModelMap model) {
        UserDAO userDAO = new UserDAO();
        HttpSession session = request.getSession();
        User loggedInUser = (User) session.getAttribute("loggedInUser");
        if (loggedInUser == null) {
            model.addAttribute("login", "Please login to follow user!");
            return "userdetails";
        }

        boolean result = userDAO.followUser(loggedInUser.getId(), userId);
        if (!result) {
            model.addAttribute("followFailure", "Follow Failed! Please try again!");
            return "userdetails";
        }

        loggedInUser.getFollowList().add(userId);
        session.setAttribute("loggedInUser", loggedInUser);
        return "redirect:/user_details?user_id=" + userId;
    }

    @RequestMapping(path = "/stopfollow", method = RequestMethod.GET)
    public String stopFollowUser(@RequestParam("user_id") int userId, HttpServletRequest request,
ModelMap model) {
        UserDAO userDAO = new UserDAO();
        HttpSession session = request.getSession();
        User loggedInUser = (User) session.getAttribute("loggedInUser");

        boolean result = userDAO.unfollowUser(loggedInUser.getId(), userId);
        if (!result) {
            model.addAttribute("removeFailure", "Remove Failed! Please try again!");
            return "userdetails";
        }

        loggedInUser.getFollowList().remove(userId);
        session.setAttribute("loggedInUser", loggedInUser);
        return "redirect:/user_details?user_id=" + userId;
```

```
    }
}
```

4. Views

(1) main.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/9
  Time: 11:16
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Welcome</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>
<div class="container">
    <br/>
    <h1 class="text-center">KSearch</h1>
    <br/>
```

```
    <form method="post" action="/search">
        <div class="form-group">
            <div class="input-group">
                <input class="form-control" type="text" name="keyword" placeholder="Enter Business
Name"/>
                <input class="btn btn-primary" type="submit" value="Search"/>
            </div>
        </div>
    </form>
</div>
</body>
</html>
```

(2) login.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/16
  Time: 13:00
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<style>
    .col-center-block {
        float: none;
        display: block;
        margin-left: auto;
        margin-right: auto;
    }
</style>

<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
</nav>

<div class="container">
    <br/>
    <h1 class="text-center">Log In to KSearch</h1>
    <h6 class="text-center">
        New to KSearch? <a href="/signup">Sign Up</a>
    </h6>
    <c:if test="${failure ne null}">
        <p class="text-center text-danger">${failure}</p>
    </c:if>
    <div class="row">
        <div class="col-xs-6 col-md-4 col-center-block">
            <br/>
            <form:form method="post" modelAttribute="userLogin">
                <div class="form-group">
                    <form:input type="text" path="email" placeholder="Enter Email" class="form-
control"/>
                    <form:errors path="email" class="text-danger"/><br/>
                    <form:input type="password" path="password" placeholder="Enter Password"
class="form-control"/>
                    <form:errors path="password" class="text-danger"/><br/>
                    <input class="btn btn-primary btn-block" type="submit" name="submit"
value="Login"/>
```

```
                </div>
            </form:form>
            <p class="text-right">New to KSearch? <a href="/signup">Sign Up</a></p>
        </div>
    </div>
</div>
</body>
</html>
```

(3) signup.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/16
  Time: 13:00
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>Sign Up</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<style>
    .col-center-block {
        float: none;
        display: block;
        margin-left: auto;
        margin-right: auto;
    }
</style>

<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
</nav>

<div class="container">
    <br/>
    <h1 class="text-center">Sign Up for KSearch</h1>
    <h6 class="text-center">Connect with great local businesses</h6>
    <c:if test="${failure ne null}">
        <p class="text-center text-danger">${failure}</p>
    </c:if>
    <div class="row">
        <div class="col-xs-6 col-md-4 col-center-block">
            <form:form method="post" modelAttribute="userSignUp">
                <div class="form-group">
                    <form:input type="text" path="firstName" placeholder="First Name" class="form-
control"/>
                    <form:errors path="firstName" class="text-danger"/><br/>
                    <form:input type="text" path="lastName" placeholder="Last Name" class="form-
control"/>
                    <form:errors path="lastName" class="text-danger"/><br/>
                    <form:input type="text" path="email" placeholder="Email" class="form-control"/>
                    <form:errors path="email" class="text-danger"/><br/>
                    <form:input type="password" path="password" placeholder="Password" class="form-
control"/>
                    <form:errors path="password" class="text-danger"/><br/>
                    <input class="btn btn-primary btn-block" type="submit" name="submit"
value="Sign Up"/><br/>
                </div>
            </form:form>
```

```
            <p class="text-right">Already On KSearch? <a href="/login">Log in</a></p>
        </div>
    </div>
</div>

</body>
</html>
```

(4) search.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/18
  Time: 14:40
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Search Result</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>

<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>
```

```
<div class="container">
    <br/>
    <form method="post" action="/search">
        <div class="form-group">
            <div class="input-group">
                <input class="form-control" type="text" name="keyword" value="${keyword}"
placeholder="Enter Business Name"/>
                <input class="btn btn-primary" type="submit" value="Search"/>
            </div>
        </div>
    </form>
</div>

<div class="container">
    <div class="border border-top-0 border-left-0 border-right-0">
        <h5>${resultCount} results found</h5>
    </div>
    <c:forEach var="business" items="${searchResult}">
        <div class="container border border-top-0 border-left-0 border-right-0">
            <h4><a href="/business/${business.id}">${business.name}</a></h4>
            <div class="row">
                <div class="col-sm-4">
                    <p>
                        <b>Stars: </b>${business.stars}<br/>
                            ${business.reviewCount} reviews<br/>
                        <c:forEach var="category" items="${business.categories}"
varStatus="status">
                            <c:choose>
                                <c:when test="${status.last eq false}">
                                    ${category},
                                </c:when>
                                <c:otherwise>
                                    ${category}
                                </c:otherwise>
                            </c:choose>
                        </c:forEach>
                    </p>
                </div>
                <div class="col-sm-8">
                    <p>
                        <b>Address: </b><br/>
                            ${business.address}<br/>
                            ${business.city}, ${business.state} ${business.postalCode}
                    </p>
                </div>
            </div>
        </div>
    </c:forEach>
    <br/>
    <ul class="pagination">
        <c:choose>
            <c:when test="${currentPage > 1}">
                <li class="page-item">
                    <a class="page-link" href="/search?keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                </li>
            </c:when>
            <c:otherwise>
                <li class="page-item disabled">
                    <a class="page-link" href="/search?keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                </li>
            </c:otherwise>
        </c:choose>
        <c:forEach var="i" begin="1" end="${pageCount}">
            <c:choose>
                <c:when test="${i == currentPage}">
```

```
                    <li class="page-item active">
                        <a class="page-link" href="/search?keyword=${keyword}&page=${i}">${i}</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item">
                        <a class="page-link" href="/search?keyword=${keyword}&page=${i}">${i}</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </c:forEach>
        <c:choose>
            <c:when test="${currentPage < pageCount}">
                <li class="page-item">
                    <a class="page-link" href="/search?keyword=${keyword}&page=${currentPage +
1}">Next</a>
                </li>
            </c:when>
            <c:otherwise>
                <li class="page-item disabled">
                    <a class="page-link" href="/search?keyword=${keyword}&page=${currentPage +
1}">Next</a>
                </li>
            </c:otherwise>
        </c:choose>
    </ul>
</div>
</body>
</html>
```

(5) business.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/18
  Time: 18:00
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>${business.name}</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
```

```
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="container border border-top-0 border-left-0 border-right-0">
    <br/>
    <h1>${business.name}</h1>
    <p><b>Stars: </b>${business.stars}</p>
    <p>${business.reviewCount} reviews</p>
    <p>
        <b>Categories:</b>
        <c:forEach var="category" items="${business.categories}" varStatus="status">
            <c:choose>
                <c:when test="${status.last eq false}">
                    ${category},
                </c:when>
                <c:otherwise>
                    ${category}
                </c:otherwise>
            </c:choose>
        </c:forEach>
    </p>
    <a href="/writeareview/business/${business.id}" class="btn btn-primary">Write a Review</a>
    <p></p>
</div>

<div class="container border border-top-0 border-left-0 border-right-0">
    <br/>
    <h3>Location & Hours</h3>
    <div class="row">
        <div class="col">
            <h5><b>Location</b></h5>
            <p>${business.address}<br/>${business.city}, ${business.state}
${business.postalCode}</p>
        </div>
        <div class="col">
            <h5><b>Hours</b></h5>
            <table>
                <c:forEach var="hours" items="${business.hours}">
                    <tr/>
                    <td/><b>${hours.day}</b>
                    <td/>${hours.startTime} - ${hours.closeTime}
                </c:forEach>
            </table>
        </div>
    </div>
    <br/>
</div>

<div class="container border border-top-0 border-left-0 border-right-0">
```

```
    <br/>
    <h3>Known For</h3>
    <div class="list-group">
        <c:forEach var="attribute" items="${business.attributes}">
            <p><b>${attribute.name}</b>: ${attribute.value}</p>
        </c:forEach>
    </div>
    <br/>
</div>

<div class="container">
    <br/>
    <h3>Reviews</h3>
    <br/>
    <div class="container">
        <form method="post" action="/business/${business.id}">
            <div class="form-group">
                <div class="input-group">
                    <input class="form-control" type="text" name="keyword" value="${keyword}"
placeholder="Search within the reviews"/>
                    <input class="btn btn-primary" type="submit" value="Search"/>
                </div>
            </div>
        </form>
    </div>

    <div class="container">
        <c:if test="${not empty keyword}">
            <div class="border border-top-0 border-left-0 border-right-0">
                <h5>${resultCount} reviews mentioning "${keyword}"</h5>
            </div>
        </c:if>
        <c:forEach var="review" items="${reviews}">
            <div class="row border border-top-0 border-left-0 border-right-0">
                <div class="col-sm-3">
                    <h6>
                        <b>User Name: </b>
                        <a href="/user_details?user_id=${review.user.id}">${review.user.name}</a>
                    </h6>
                    <p>
                        ${review.user.followList.size()} friends<br/>
                        ${review.user.reviews.size()} reviews
                    </p>
                </div>
                <div class="col-sm-9">
                    <p><b>Stars: </b>${review.stars}</p>
                    <p><b>Date: </b>${review.date}</p>
                    <p>${review.text}</p>
                </div>
            </div>
        </c:forEach>
    </div>
    <br/>
    <ul class="pagination">
        <c:choose>
            <c:when test="${currentPage > 1}">
                <li class="page-item">
                    <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                </li>
            </c:when>
            <c:otherwise>
                <li class="page-item disabled">
                    <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                </li>
```

```
            </c:otherwise>
        </c:choose>
        <c:forEach var="i" begin="1" end="${pageCount}">
            <c:choose>
                <c:when test="${i == currentPage}">
                    <li class="page-item active">
                        <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${i}">${i}</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item">
                        <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${i}">${i}</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </c:forEach>
        <c:choose>
            <c:when test="${currentPage < pageCount}">
                <li class="page-item">
                    <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                </li>
            </c:when>
            <c:otherwise>
                <li class="page-item disabled">
                    <a class="page-link"
href="/business/${business.id}/search_review?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                </li>
            </c:otherwise>
        </c:choose>
    </ul>
</div>
</body>
</html>
```

(6) writeareview.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/19
  Time: 1:10
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Write A Review</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item active">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>
```

```
        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="jumbotron">
    <div class="container">
        <h1>Your First Review Awaits</h1>
        <h4>Review everything from your favorite burger to your dentist.</h4>
        <br/>
        <form method="post" action="/writeareview/search">
            <div class="form-group">
                <div class="input-group">
                    <input class="form-control" type="text" name="keyword" value="${keyword}"
placeholder="Enter Business Name"/>
                    <input class="btn btn-primary" type="submit" value="Search"/>
                </div>
            </div>
        </form>
    </div>
</div>

<c:if test="${keyword ne null}">
    <div class="container">
        <div class="border border-top-0 border-left-0 border-right-0">
            <h3>See the business you'd like to review?</h3>
        </div>

        <c:forEach var="business" items="${searchResult}">
            <div class="container border border-top-0 border-left-0 border-right-0">
                <h5>${business.name}</h5>
                <div class="row">
                    <div class="col-sm-4">
                        <p>
                            <b>Stars: </b>${business.stars}<br/>
                                ${business.reviewCount} reviews<br/>
                            <c:forEach var="category" items="${business.categories}"
varStatus="status">
                                <c:choose>
                                    <c:when test="${status.last eq false}">
                                        ${category},
                                    </c:when>
                                    <c:otherwise>
                                        ${category}
```

```
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                        </p>
                    </div>
                    <div class="col-sm-8">
                        <p>
                            <b>Address: </b><br/>
                                ${business.address}<br/>
                                ${business.city}, ${business.state} ${business.postalCode}
                        </p>
                    </div>
                </div>
                <a href="/writeareview/business/${business.id}" class="btn btn-primary">Write a
Review</a>
                <p></p>
            </div>
        </c:forEach>

        <ul class="pagination">
            <c:choose>
                <c:when test="${currentPage > 1}">
                    <li class="page-item">
                        <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${currentPage - 1}">Previous</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item disabled">
                        <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${currentPage - 1}">Previous</a>
                    </li>
                </c:otherwise>
            </c:choose>
            <c:forEach var="i" begin="1" end="${pageCount}">
                <c:choose>
                    <c:when test="${i == currentPage}">
                        <li class="page-item active">
                            <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${i}">${i}</a>
                        </li>
                    </c:when>
                    <c:otherwise>
                        <li class="page-item">
                            <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${i}">${i}</a>
                        </li>
                    </c:otherwise>
                </c:choose>
            </c:forEach>
            <c:choose>
                <c:when test="${currentPage < pageCount}">
                    <li class="page-item">
                        <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item disabled">
                        <a class="page-link"
href="/writeareview/search?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</c:if>
```

```
</body>
</html>
```

(7) rate.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/19
  Time: 2:17
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Write a Review</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item active">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="container">
    <h1><a href="/business/${business.id}?page=1">${business.name}</a></h1>
    <c:if test="${failure}">
        <p class="text-danger">Please <a href="/login">login</a> to post your review!</p>
```

```
    </c:if>
    <c:if test="${postFailure}">
        <p class="text-danger">Post Review Failed! Please try again later!</p>
    </c:if>

    <form:form method="post" modelAttribute="starsComment">
        <div class="form-group border">
            <div class="row">
                <div class="col-sm-3">
                    <form:select path="stars" class="form-control">
                        <option selected>5</option>
                        <option>4</option>
                        <option>3</option>
                        <option>2</option>
                        <option>1</option>
                    </form:select>
                </div>
            </div>
            <br/>
            <form:textarea path="text" rows="15" placeholder="Enter your comment here" class="form-
control"/>
            <form:errors path="text" class="text-danger"/>
        </div>
        <input class="btn btn-primary btn-block" type="submit" name="submit" value="Post Review"/>
    </form:form>
</div>

</body>
</html>
```

(8) userdetails.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/22
  Time: 4:23
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>${user.name}'s Reviews</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
```

```
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="jumbotron">
    <div class="container">
        <div class="row">
            <div class="col-sm-9">
                <h1>${user.name}</h1>
                <p>
                    ${user.followList.size()} friends
                    ${user.reviewCount} reviews
                </p>
                <p>
                    <b>Elite Years: </b>
                    <c:choose>
                        <c:when test="${user.elite.size() == 0}">
                            None
                        </c:when>
                        <c:otherwise>
                            <c:forEach var="year" items="${user.elite}" varStatus="status">
                                <c:choose>
                                    <c:when test="${status.last eq false}">
                                        ${year},
                                    </c:when>
                                    <c:otherwise>
                                        ${year}
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                        </c:otherwise>
                    </c:choose>
                </p>
            </div>
            <div class="col-sm-3">
                <c:choose>
                    <c:when test="${loggedInUser ne null and user.id == loggedInUser.id}">
                        <a href="/profile">Update Your Profile</a><br/>
                        <a href="/search_friends">Find Friends</a>
                    </c:when>
                    <c:when test="${loggedInUser eq null}">

                    </c:when>
                    <c:otherwise>
                        <c:choose>
                            <c:when test="${loggedInUser.followList.contains(user.id)}">
                                <a href="/stopfollow?user_id=${user.id}">Stop Follow
${user.name}</a><br/>
                                <c:if test="${removeFailure ne null}">
                                    ${removeFailure}<br/>
```

```
                                </c:if>
                            </c:when>
                            <c:otherwise>
                                <a href="/follow?user_id=${user.id}">Follow ${user.name}</a><br/>
                                <c:if test="${login ne null}">
                                    ${login}<br/>
                                </c:if>
                                <c:if test="${followFailure ne null}">
                                    ${followFailure}<br/>
                                </c:if>
                            </c:otherwise>
                        </c:choose>
                    </c:otherwise>
                </c:choose>
            </div>
        </div>
    </div>
</div>

<div class="container">
    <div class="row">
        <div class="col-sm-3">
            <h4 class="text-center">${user.name}'s Profile</h4><br/>
            <div class="list-group bg-light">
                <a href="/user_details?user_id=${user.id}" class="bg-light list-group-item list-
group-item-action border-left-0 border-right-0">
                    Profile Overview
                </a>
                <a href="/user_details/followers?user_id=${user.id}" class="bg-light list-group-
item list-group-item-action border-left-0 border-right-0">
                    Followers
                </a>
                <a href="/user_details/follows?user_id=${user.id}" class="bg-light list-group-item
list-group-item-action border-left-0 border-right-0">
                    Followed
                </a>
                <a href="/user_details/reviews?user_id=${user.id}" class="bg-light list-group-item
list-group-item-action border-left-0 border-right-0">
                    Reviews
                </a>
            </div>
        </div>
        <div class="col-sm-6 border border-top-0 border-left-0 border-bottom-0">
            <c:if test="${showReviews ne null}">
                <div class="border border-top-0 border-left-0 border-right-0">
                    <h3>Reviews</h3>
                </div>
                <c:choose>
                    <c:when test="${user.reviewCount == 0}">
                        <p>${user.name} hasn't written any reviews just yet.</p>
                    </c:when>
                    <c:otherwise>
                        <c:forEach var="review" items="${showReviews}">
                            <div class="border border-top-0 border-left-0 border-right-0">
                                <h5><a
href="/business/${review.business.id}">${review.business.name}</a></h5>
                                <p>
                                    <b>Categories:</b>
                                    <c:forEach var="category" items="${review.business.categories}"
varStatus="status">
                                        <c:choose>
                                            <c:when test="${status.last eq false}">
                                                ${category},
                                            </c:when>
                                            <c:otherwise>
                                                ${category}<br/>
                                            </c:otherwise>
                                        </c:choose>
```

```
                                    </c:forEach>
                                        ${review.business.address}<br/>
                                        ${review.business.city}, ${review.business.state}
${review.business.postalCode}<br/>
                                    <b>Stars: </b>${review.stars}, ${review.date}
                                </p>
                                <p>${review.text}</p>
                            </div>
                        </c:forEach>
                        <a href="/user_details/reviews?user_id=${user.id}">View All Reviews</a>
                    </c:otherwise>
                </c:choose>
            </c:if>
            <c:if test="${followerList ne null}">
                <div class="border border-top-0 border-left-0 border-right-0">
                    <h3>Followers</h3>
                </div>
                <c:choose>
                    <c:when test="${followerList.size() == 0}">
                        <p>${user.name} doesn't have any followers yet.</p>
                    </c:when>
                    <c:otherwise>
                        <c:forEach var="follower" items="${followerList}">
                            <div class="border border-top-0 border-left-0 border-right-0">
                                <h5><a
href="/user_details?user_id=${follower.id}">${follower.name}</a></h5>
                                <p>
                                    ${follower.followList.size()} friends<br/>
                                    ${follower.reviews.size()} reviews<br/>
                                    Elite Years:
                                    <c:choose>
                                        <c:when test="${follower.elite.size() == 0}">
                                            None
                                        </c:when>
                                        <c:otherwise>
                                            <c:forEach var="year" items="${follower.elite}"
varStatus="status">
                                                <c:choose>
                                                    <c:when test="${status.last eq false}">
                                                        ${year},
                                                    </c:when>
                                                    <c:otherwise>
                                                        ${year}
                                                    </c:otherwise>
                                                </c:choose>
                                            </c:forEach>
                                        </c:otherwise>
                                    </c:choose>
                                </p>
                            </div>
                        </c:forEach>
                        <ul class="pagination">
                            <c:choose>
                                <c:when test="${currentPage > 1}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${currentPage - 1}">Previous</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${currentPage - 1}">Previous</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                            <c:forEach var="i" begin="1" end="${pageCount}">
                                <c:choose>
```

```
                                    <c:when test="${i == currentPage}">
                                        <li class="page-item active">
                                            <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${i}">${i}</a>
                                        </li>
                                    </c:when>
                                    <c:otherwise>
                                        <li class="page-item">
                                            <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${i}">${i}</a>
                                        </li>
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                            <c:choose>
                                <c:when test="${currentPage < pageCount}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${currentPage + 1}">Next</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/followers_pagination?user_id=${user.id}&page=${currentPage + 1}">Next</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                        </ul>
                    </c:otherwise>
                </c:choose>
            </c:if>
            <c:if test="${followList ne null}">
                <div class="border border-top-0 border-left-0 border-right-0">
                    <h3>Followed</h3>
                </div>
                <c:choose>
                    <c:when test="${followList.size() == 0}">
                        <p>${user.name} doesn't follow any users yet.</p>
                    </c:when>
                    <c:otherwise>
                        <c:forEach var="follow" items="${followList}">
                            <div class="border border-top-0 border-left-0 border-right-0">
                                <h5><a
href="/user_details?user_id=${follow.id}">${follow.name}</a></h5>
                                <p>
                                    ${follow.followList.size()} friends<br/>
                                    ${follow.reviews.size()} reviews<br/>
                                    Elite Years:
                                    <c:choose>
                                        <c:when test="${follow.elite.size() == 0}">
                                            None
                                        </c:when>
                                        <c:otherwise>
                                            <c:forEach var="year" items="${follow.elite}"
varStatus="status">
                                                <c:choose>
                                                    <c:when test="${status.last eq false}">
                                                        ${year},
                                                    </c:when>
                                                    <c:otherwise>
                                                        ${year}
                                                    </c:otherwise>
                                                </c:choose>
                                            </c:forEach>
                                        </c:otherwise>
                                    </c:choose>
                                </p>
```

```
                            </div>
                        </c:forEach>
                        <ul class="pagination">
                            <c:choose>
                                <c:when test="${currentPage > 1}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${currentPage - 1}">Previous</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${currentPage - 1}">Previous</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                            <c:forEach var="i" begin="1" end="${pageCount}">
                                <c:choose>
                                    <c:when test="${i == currentPage}">
                                        <li class="page-item active">
                                            <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${i}">${i}</a>
                                        </li>
                                    </c:when>
                                    <c:otherwise>
                                        <li class="page-item">
                                            <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${i}">${i}</a>
                                        </li>
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                            <c:choose>
                                <c:when test="${currentPage < pageCount}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${currentPage + 1}">Next</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/follows_pagination?user_id=${user.id}&page=${currentPage + 1}">Next</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                        </ul>
                    </c:otherwise>
                </c:choose>
            </c:if>
            <c:if test="${reviewList ne null}">
                <div class="border border-top-0 border-left-0 border-right-0">
                    <h3>Reviews</h3>
                </div>
                <c:choose>
                    <c:when test="${user.reviewCount == 0}">
                        <p>${user.name} hasn't written any reviews just yet.</p>
                    </c:when>
                    <c:otherwise>
                        <br/>
                        <div class="container">
                            <form method="post" action="/user_details/reviews?user_id=${user.id}">
                                <div class="form-group">
                                    <div class="input-group">
                                        <input class="form-control" type="text" name="keyword"
value="${keyword}" placeholder="Search within the reviews"/>
                                        <input class="btn btn-primary" type="submit"
```

```
value="Search"/>
                                    </div>
                                </div>
                            </form>
                        </div>

                        <div class="container">
                            <c:if test="${not empty keyword}">
                                <div class="border border-top-0 border-left-0 border-right-0">
                                    <h5>${resultCount} reviews mentioning "${keyword}"</h5>
                                </div>
                            </c:if>
                            <c:forEach var="review" items="${reviewList}">
                                <div class="border border-top-0 border-left-0 border-right-0">
                                    <h5><a
href="/business/${review.business.id}">${review.business.name}</a></h5>
                                    <p>
                                        <b>Categories:</b>
                                        <c:forEach var="category"
items="${review.business.categories}" varStatus="status">
                                            <c:choose>
                                                <c:when test="${status.last eq false}">
                                                    ${category},
                                                </c:when>
                                                <c:otherwise>
                                                    ${category}<br/>
                                                </c:otherwise>
                                            </c:choose>
                                        </c:forEach>
                                        ${review.business.address}<br/>
                                        ${review.business.city}, ${review.business.state}
${review.business.postalCode}<br/>
                                        <b>Stars: </b>${review.stars}, ${review.date}
                                    </p>
                                    <p>${review.text}</p>
                                </div>
                            </c:forEach>
                        </div>
                        <br/>
                        <ul class="pagination">
                            <c:choose>
                                <c:when test="${currentPage > 1}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${currentPage -
1}">Previous</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                            <c:forEach var="i" begin="1" end="${pageCount}">
                                <c:choose>
                                    <c:when test="${i == currentPage}">
                                        <li class="page-item active">
                                            <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${i}">${i}</a>
                                        </li>
                                    </c:when>
                                    <c:otherwise>
                                        <li class="page-item">
                                            <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${i}">${i}</a>
```

```
                                        </li>
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                            <c:choose>
                                <c:when test="${currentPage < pageCount}">
                                    <li class="page-item">
                                        <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${currentPage +
1}">Next</a>
                                    </li>
                                </c:when>
                                <c:otherwise>
                                    <li class="page-item disabled">
                                        <a class="page-link"
href="/user_details/reviews_search?user_id=${user.id}&keyword=${keyword}&page=${currentPage +
1}">Next</a>
                                    </li>
                                </c:otherwise>
                            </c:choose>
                        </ul>
                    </c:otherwise>
                </c:choose>
            </c:if>
        </div>
        <div class="col-sm-3">
            <h3>About ${user.name}</h3>
            <div>
                <h5>Rating Distribution</h5>
                <p>
                    <b>5 Stars: </b>${ratingDistribution.get(4)}<br/>
                    <b>4 Stars: </b>${ratingDistribution.get(3)}<br/>
                    <b>3 Stars: </b>${ratingDistribution.get(2)}<br/>
                    <b>2 Stars: </b>${ratingDistribution.get(1)}<br/>
                    <b>1 Stars: </b>${ratingDistribution.get(0)}<br/>
                </p>
            </div>
            <br/>
            <div>
                <h5>Stats</h5>
                <p>
                    <b>Followers: </b>${user.followList.size()}<br/>
                    <b>Joining Since: </b>${user.dateJoined}<br/>
                </p>
            </div>
        </div>
    </div>
</div>

</body>
</html>
```

(9) profile.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/23
  Time: 0:04
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Edit Your Profile</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
```

```
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>

<style>
    .col-center-block {
        float: none;
        display: block;
        margin-left: auto;
        margin-right: auto;
    }
</style>

<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="container">
    <br/>
    <div class="border border-top-0 border-left-0 border-right-0">
        <h1>Profile</h1>
    </div>
    <c:if test="${failure ne null}">
        <p class="text-center text-danger">${failure}</p>
    </c:if>
    <br/>
    <div class="row">
        <div class="col-xs-6 col-md-4 col-center-block">
            <form:form method="post" modelAttribute="userSignUp">
                <div class="form-group">
```

```
                    <form:input type="text" path="firstName" placeholder="First Name" class="form-
control"/>
                    <form:errors path="firstName" class="text-danger"/><br/>
                    <form:input type="text" path="lastName" placeholder="Last Name" class="form-
control"/>
                    <form:errors path="lastName" class="text-danger"/><br/>
                    <form:input type="text" path="email" placeholder="Email" class="form-control"/>
                    <form:errors path="email" class="text-danger"/><br/>
                    <form:input type="password" path="password" placeholder="Password" class="form-
control"/>
                    <form:errors path="password" class="text-danger"/><br/>
                    <input class="btn btn-primary btn-block" type="submit" name="submit"
value="Save Changes"/><br/>
                </div>
            </form:form>
        </div>
    </div>
</div>

</body>
</html>
```

(10) findfriends.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/23
  Time: 1:17
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:set var="loggedInUser" value="${sessionScope.loggedInUser}"/>
<html>
<head>
    <title>Find Friends</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/popper.js/1.12.5/umd/popper.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/4.1.0/js/bootstrap.min.js"></script>
</head>
<body class="bg-light">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="navbar-header">
        <a class="navbar-brand" href="/main">KSearch</a>
    </div>
    <div class="container-fluid">
        <ul class="nav navbar-nav">
            <li class="nav-item">
                <a class="nav-link" href="/writeareview">Write A Review</a>
            </li>
        </ul>

        <ul class="nav navbar-nav">
            <c:choose>
                <c:when test="${loggedInUser ne null}">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
                            Hello, ${loggedInUser.name}!
                        </a>
                        <div class="dropdown-menu">
                            <a class="dropdown-item"
href="/user_details?user_id=${loggedInUser.id}">About Me</a>
                            <a class="dropdown-item" href="/profile">Account Settings</a>
                            <a class="dropdown-item" href="/logout">Log Out</a>
                        </div>
                    </li>
```

```
                </c:when>
                <c:otherwise>
                    <li class="nav-item">
                        <a class="nav-link" href="/login">Log In</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/signup">Sign Up</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</nav>

<div class="jumbotron">
    <div class="container">
        <h1>Search Friends on KSearch</h1>
        <br/>
        <form method="post" action="/find_friends">
            <div class="form-group">
                <div class="input-group">
                    <input class="form-control" type="text" name="keyword" value="${keyword}"
placeholder="Enter User Name"/>
                    <input class="btn btn-primary" type="submit" value="Search"/>
                </div>
            </div>
        </form>
    </div>
</div>

<c:if test="${keyword ne null}">
    <div class="container">
        <div class="border border-top-0 border-left-0 border-right-0">
            <h3>Member Search Results</h3>
        </div>

        <c:forEach var="user" items="${searchResult}">
            <div class="container border border-top-0 border-left-0 border-right-0">
                <h5><a href="/user_details?user_id=${user.id}">${user.name}</a></h5>
                <p>
                    ${user.followList.size()} friends<br/>
                    ${user.reviewCount} reviews<br/>
                    <b>Elite Years: </b>
                    <c:choose>
                        <c:when test="${user.elite.size() == 0}">
                            None
                        </c:when>
                        <c:otherwise>
                            <c:forEach var="year" items="${user.elite}" varStatus="status">
                                <c:choose>
                                    <c:when test="${status.last eq false}">
                                        ${year},
                                    </c:when>
                                    <c:otherwise>
                                        ${year}
                                    </c:otherwise>
                                </c:choose>
                            </c:forEach>
                        </c:otherwise>
                    </c:choose>
                </p>
            </div>
        </c:forEach>

        <ul class="pagination">
            <c:choose>
                <c:when test="${currentPage > 1}">
                    <li class="page-item">
```

```
                        <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${currentPage - 1}">Previous</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item disabled">
                        <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${currentPage - 1}">Previous</a>
                    </li>
                </c:otherwise>
            </c:choose>
            <c:forEach var="i" begin="1" end="${pageCount}">
                <c:choose>
                    <c:when test="${i == currentPage}">
                        <li class="page-item active">
                            <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${i}">${i}</a>
                        </li>
                    </c:when>
                    <c:otherwise>
                        <li class="page-item">
                            <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${i}">${i}</a>
                        </li>
                    </c:otherwise>
                </c:choose>
            </c:forEach>
            <c:choose>
                <c:when test="${currentPage < pageCount}">
                    <li class="page-item">
                        <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                    </li>
                </c:when>
                <c:otherwise>
                    <li class="page-item disabled">
                        <a class="page-link"
href="/find_friends?keyword=${keyword}&page=${currentPage + 1}">Next</a>
                    </li>
                </c:otherwise>
            </c:choose>
        </ul>
    </div>
</c:if>



</body>
</html>
```